IPS: Instance Profile for Shapelet Discovery for Time Series Classification

Guozhong Li*, Byron Choi*, Jianliang Xu*, Sourav S Bhowmick†, Daphne Ngar-yin Mah‡ and Grace L.H. Wong§
*Department of Computer Science, Hong Kong Baptist University, Hong Kong SAR
† School of Computing Engineering, Nanyang Technological University, Singapore
‡ Department of Geography, Hong Kong Baptist University, Hong Kong SAR
§Department of Medicine & Therapeutics, The Chinese University of Hong Kong, Hong Kong SAR
*{csgzli,bchoi,xujl}@comp.hkbu.edu.hk, †assourav@ntu.edu.sg, ‡daphnemah@hkbu.edu.hk, §wonglaihung@cuhk.edu.hk

***Abstract*—Time series classification (TSC) has been one of the most fundamental problems of time series data. Time series shapelets (or simply, *shapelets*) are discriminative subsequences that have been recently found both effective and interpretable for solving TSC. However, shapelet discovery is known to be computationally costly. Meanwhile, matrix profile has been recently proposed for efficient motif discovery and anomaly detection. Our preliminary experiment shows that a direct adoption of the matrix profile on TSC does not bring superior classification accuracy. We have identified two main issues of such an adoption: 1) discords as "shapelets", and 2) lack of shapelet diversity. In response to these issues, we propose instance profile for shapelets, called IPS, for shapelet discovery for TSC. The main challenge is to utilize the instance profile (IP) to capture the characteristics of shapelets in a robust manner and then to discover high-quality shapelets efficiently. First, we use our IP to generate abundant shapelet candidates. We next efficiently prune candidates that do not align with the definition of shapelets using a novel distribution-aware bloom filter (DABF). Three utility functions are proposed to measure the shapelet candidates and DABF is used to efficiently compute the functions. We have conducted comprehensive experiments on IPS with 12 competitive state-of-the-art methods using UCR Archive datasets. The efficiency is on average 25 times faster than that of BSPCOVER (the current state-of-the-art method). The accuracy of IPS is comparable to or higher than that of existing work. Furthermore, we select one case study to illustrate the interpretability of the shapelets.**

***Index Terms*—Time series classification, Instance profile, Distribution-aware bloom filter, Efficiency, Accuracy**

I. Introduction

Time series classification (TSC) is one of the most fundamental analyses of time series data. TSC has attracted substantial research attention, such as [2], [12], [13], [21], [31], [33]. The traditional approaches to solving the TSC problem can be classified into four categories, namely, whole series-based, intervals-based, dictionary-based, and model-based [2]. Raw data analysis, feature extraction, frequency of subsequences' repetition, and generative model, among other detailed methods, have been employed in these approaches. Recently, Ye et al. [35] have proposed shapelets, which can be intuitively considered as discriminative subsequences that maximally represent classes of the time series. Shapelet-based methods (*e.g.*, [11], [16], [23], [26], [35]) have repeatedly found superior in solving TSC. Interested readers may refer to an excellent survey on TSC [2].

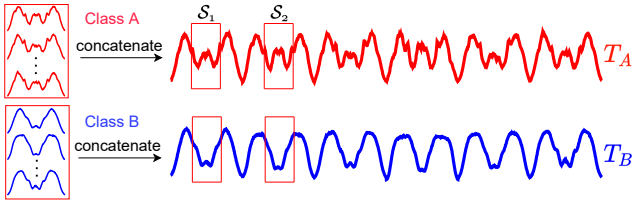


Fig. 1. Concatenations of time series (T_A and T_B) of two classes (namely, class A and class B) taken from the ArrowHead dataset of UCR Archive [9]

Fig. 2. Two shapelets of Class A for ArrowHead [9] in Figure 1, which exist in Class A but not in Class B. $\mathcal{S}_1$ and $\mathcal{S}_2$ correspond to the shapelets at the bottom of the arrow.

Shapelets are discriminative time series subsequences originally proposed for TSC [35]. Shapelets can be readily interpreted by humans as they are subsequences themselves. We illustrate an example of a popular time series dataset, called ArrowHead [9]. The left-hand side of Figure 1 shows the raw data, whereas Figure 2 shows two shapelets of Class A. A key step of shapelet-based methods is to *discover high-quality shapelets for* TSC. Lines et al. [26] propose shapelet transformation for discovering shapelets to represent the original time series into a new space. Some classic classification methods, such as Nearest Neighbor, Naive Bayes, and SVM, then can be applied to the transformed data to do classification. Grabocka et al. [16] introduce a logistic regression function to learn near-to-optimal time series shapelets for TSC, called LTS. The classification accuracy of the UCR Archive [9], a well-known benchmark of time series datasets, is thus significantly increased. Li et al. propose ShapeNet, a neural-network approach for discovering shapelets on multivariate TSC [24]. However, the efficiency of shapelet discovery has not been the main focus of these previous works. The number of shapelet candidates is large, and a few parameters, such as shapelet number and length, are explored manually for TSC.

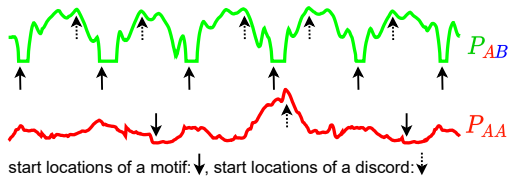


Fig. 3. The matrix profiles of (T_A, T_B) and (T_A, T_A) of Figure 1, denoted as P_{AB} and P_{AA}

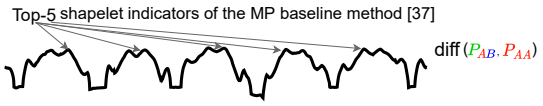


Fig. 4. The difference between the matrix profile P_{AB} and P_{AA} of Figure 3 denoted as $\mathsf{diff}(P_{AB}, P_{AA})$

Recently, matrix profile (MP), a data structure to annotate time series, has been proposed for computing the time series motifs and discords. One strength of matrix profile is to discover the time series motifs and discords both *efficiently* and *accurately* [1], [8], [36], [37]. However, few matrix profile-based shapelet methods for TSC have been proposed. Yeh et al. [37] sketched a baseline method, concatenating all the time series instances of each class (*e.g.*, Class A and B in Figure 1), and then computing the corresponding matrix profile, as shown in Figure 3. It has been argued [37] that the large differences (in Figure 4) of the matrix profiles can be indicators of high-quality shapelet candidates. The intuition is that a discriminative pattern presenting in, say, Class A, but not Class B, results in a big gap in the values between the matrix profiles.

There are however two main issues of such a method for discovering shapelets, which can yield detrimental effects on the quality of the shapelets (to be detailed in Section II-B). In a nutshell, the indicator may identify a discord of both Class A and Class B as a shapelet (1st issue, discords as "shapelets"), which is possible to have a large difference between the matrix profiles. This situation contradicts the definition of shapelets, which are generally close to one class but far away from the other classes [35]. Previous work [37] concatenates all time series instances into one long instance, which reduces the diversity of shapelets (the 2nd issue, lack of shapelet diversity), although this could be mitigated by picking top-k shapelets [37]. However, the top-k shapelets can be similar to each other and the 2nd issue still exists.

Contributions. This is *the first work to take the advantage of both shapelet and matrix profile* for TSC, called *instance profile* for shapelets, namely IPS, which addresses the above-mentioned issues. An overview of IPS is presented in Figure 5.

First, we propose the instance profile (IP) on the time series instances generated from a sampling method [5], which is different from the previous work [37] (the MP baseline) to concatenate all the time series instances into one. In particular, some time series instances are randomly selected from the training set to concatenate into one long time series for the shapelet candidates, which is more robust than the MP baseline in each sample of instances. We calculate the initial motif and discord candidates. Since the candidates are voluminous, we propose distribution-aware bloom filter (DABF) to prune them in $O(N)$, where N is the length of time series. DABF can efficiently prune similar candidates via using LSH [14] and the distribution of the time series subsequences in the codomain. It is worth mentioning that the MP baseline method [37] can be viewed as an extreme case of our IPS to concatenate all the time series instances in one sample without the DABF for optimizing the efficiency.

Second, we propose three utility functions to score the shapelet candidates from three different perspectives derived from the definition of shapelets, namely *the intra class*, *inter class*, and *time series instances* from the same class. As the name suggests, the intra class utility quantifies the distance of motif candidates from the same class. The motifs and discords from the inter classes are utilized to eliminate the shapelet candidates with low quality. The time series instances from the same class can furtherly contribute to the quality of the final shapelets. We further propose two optimization techniques, namely distribution transformation (DT) and computation reuse (CR), which exploit DABF for efficiency. For self-containedness, we adopt shapelet transformation rather than directly use shapelets to do the classification [16], [23].

Finally, we conduct extensive experiments on the UCR time series archive [9]. The comparison with the MP baseline and the current state-of-the-art BSPCOVER proves the efficiency superiority of IPS. We show the efficiency of the DABF and two optimization techniques. We note that IPS performs the best accuracy in 9 datasets out of 46 datasets. IPS is ranked $4th$ on accuracy among all the 13 state-of-the-art methods. We illustrate one case study to use the discovered shapelets to interpret the classification results.

Organization. The rest of this paper is organized as follows. Section II presents some terminologies and investigates two issues with the MP baseline method. The details of our IPS method are introduced in Section III. Section IV reports the experimental results on efficiency and accuracy. Section V reviews the related work. Section VI concludes the paper.

II. Preliminaries

In this section, we introduce some background, problem statement, and the analysis of two issues with the MP baseline method for TSC.

A. Terminologies and Notations

In this subsection, we present some terminologies and summarize the notations and their meanings in Table I.

Definition 1: **Time series** T. A time series T is an ordered-value sequence $T = (t_1, t_2, \cdots, t_i, \cdots, t_N)$, where N is the length of T, $t_i \in \mathbb{R}, i \in [1, \cdots, N]$. □

Definition 2: **Time series dataset** D. A time series dataset D is a set of time series. T_j is a time series. C_j is the class label of T_j, where $C_j \in \mathcal{C}, j \in [1, M]$. $\mathcal{C} = \{1, 2, \cdots, |\mathcal{C}|\}$, and $|\mathcal{C}|$ denotes the number of classes. □

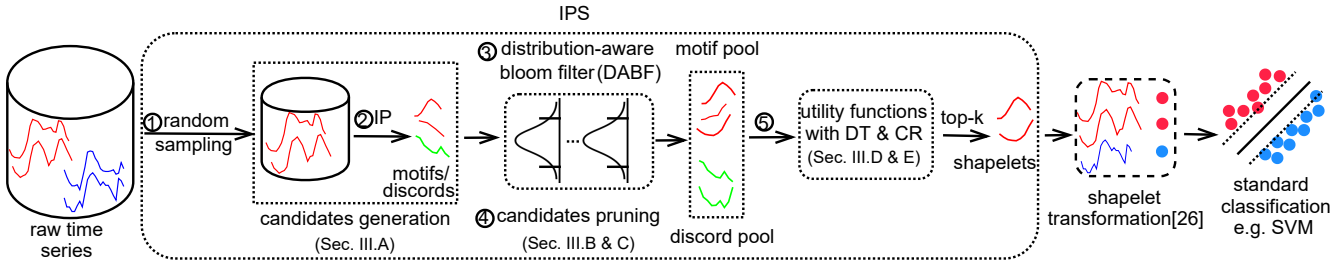


Fig. 5. The overview of IPS for TSC. From ① the multiple samples randomly selected from all the time series instances, we employ the motifs calculated by ② the instance profile (IP) as the initial shapelet candidates. The motifs are the frequent time series subsequences, which meet the part of shapelets' definition, widely existing in a dataset. However, the motifs may not only exist in one class. We then discover the motifs existing in one class through ③ the distribution-aware bloom filter (DABF) to ④ prune similar candidates from the inter (other) classes. We propose ⑤ three utility functions (from intra class motifs, inter classes motifs/discords, and intra class time series instances) to score the shapelet candidates for computing the final top-k shapelets with two optimization techniques (DT & CR) based on DABF.

TABLE I
SUMMARY OF FREQUENTLY USED NOTATIONS

Notation	Meaning
T	a time series $(t_1, t_2, \cdots, t_i, \cdots, t_N)$, where t_i is the i-th value in T and N is the length of T
$T_{a,b}$	a subsequence $T_{a,b}$ of T, $(t_a, \cdots, t_b)$, where $1 \leq a \leq b \leq N$, a and b, the beginning and ending positions of the subsequence
D	a time series dataset $(T_1, T_2, \cdots, T_M)$, where M is the number of time series in D
$\mathcal{C}$	the label set, a.k.a class $\{1, 2, \cdots, \lvert\mathcal{C}\rvert\}$
D_C	C is the label in $\mathcal{C}$; and for all $T \in D_C$, the label of T is C
T_C	concatenating all time series instances from class C into one longer time series (*e.g.*, T_A, T_B in Figure 1)
MP	the matrix profile (value)
$\mathcal{S}$	set of shapelets
diff	the difference between matrix profiles
dist	the ED distance between time series (subsequences)
label	the label of time series

Definition 3: **Subsequence** $T_{a,b}$. Given a time series T, a subsequence $T_{a,b}$ of T is $(t_a, \cdots, t_b)$, where $1 \leq a \leq b \leq N$, a and b are, respectively, the beginning and ending positions of the subsequence in T. □

Definition 4: **Distance between two time series** [11]. The distance of the sequence T_p of the length $|T_p|$ and T_q of the length $|T_q|$ is denoted as (w.l.o.g. assuming $|T_q| \geq |T_p|$),

$$\mathsf{dist}(T_p, T_q) = \min_{j=1,\cdots,|T_q|-|T_p|+1} \frac{1}{|T_p|} \sum_{l=1}^{|T_p|} (tq_{j+l-1} - tp_l)^2, \quad (1)$$

where tq_i and tp_i are the i-th value of T_p and T_q, respectively. □

Definition 5: **Matrix profile**. Given a time series T with a length N and a specific subsequence length L, the *matrix profile* is a data structure to annotate T,

$$MP(T, L) = (mp_1, mp_2, \cdots, mp_i, \cdots, mp_{N-L+1}) \quad (2)$$

where mp_i, $i \in [1, N-L+1]$ is the nearest neighbor distance of the subsequence $T_{i,i+L-1}$ among the other subsequences with length L in T.[1] □

[1]Since the subsequences located near $T_{i,i+L-1}$ may have small distance from $T_{i,i+L-1}$, those subsequences are excluded from finding mp_i.

Definition 6: **Shapelet** $\mathcal{S}$ [35]. A shapelet $\mathcal{S}$ of the length $|\mathcal{S}|$ of class C_j, where $C_j \in \mathcal{C}$, is a time series subsequence, which represents class C_j and discriminates C_j from other classes, *i.e.*, $\mathcal{C} \setminus \{C_j\}$. That is, for all T_j having the label C_j, $\mathsf{dist}(T_j, \mathcal{S})$ is smaller than $\mathsf{dist}(T_i, \mathcal{S})$, where T_i is time series having a label in $\mathcal{C} \setminus \{C_j\}$. □

According to Formula 1, we can define the distance between the j-th time series T_j and a shapelet candidate $\mathcal{S}_i$ of the length $|\mathcal{S}_i|$ as follows:

$$d_{j,i} = \mathsf{dist}(T_j, \mathcal{S}_i) \quad (3)$$

Definition 7: **Shapelet transformation** [26]. Shapelet transformation is a method to transform a time series T_j into an embedding in new data space $(d_{j,1}, \cdots, d_{j,|\mathcal{S}|})$ by calculating the distances with a set of discovered shapelets $\mathcal{S}$ for TSC, where $d_{j,i} = \mathsf{dist}(T_j, S_i)$ and $S_i \in \mathcal{S}$. □

Problem statement. Given a time series dataset D, this paper investigates an efficient shapelet discovery method for time series classification. □

B. Two issues of the MP baseline method

In this subsection, we analyze two inherent issues of the MP baseline method in detail.

Baseline. We introduce the main idea of the baseline method of directly using MP to determine shapelets [37]. The baseline selects the time series subsequence at the location of the largest difference in the MP as the final shapelet for Class A (indicated in Figure 4). Specifically, Formula 4 shows how is the subsequence selected by the baseline method. It finds a time series subsequence $\mathcal{S}$ as the "shapelet" with the largest difference between the distances of $\mathcal{S}$ and the concatenated time series T_A and T_B, respectively.

$$\arg\max(\mathsf{diff}(\mathrm{P}_{AB}, \mathrm{P}_{AA})) = \arg\max_{\substack{\mathcal{S} \\ \mathsf{label}(T_B) \neq \mathsf{label}(T_A)}} ||\mathsf{dist}(\mathcal{S}, T_B) - \mathsf{dist}(\mathcal{S}, T_A)||, \quad (4)$$

where $\mathcal{S}$ is from Class A of length L, starting from $i \in [0, N-L+1]$.

The idea above can be easily extended to discover the top-k shapelets by using the top-k largest differences. We then

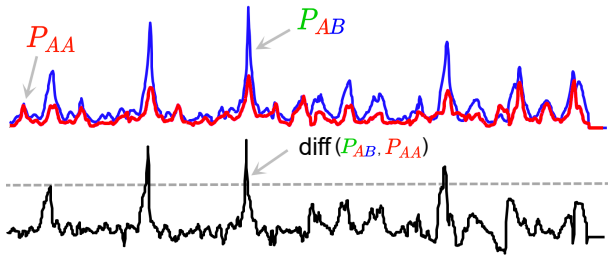


Fig. 6. An example of matrix profiles and the difference to show the 1st issue Discords as "shapelets" [37]. We observe the largest difference in the below part. The value in the P_{AB} (the above part) is the largest one, the value in the P_{AA} (the above part) is however the largest one (the second scenario). That value in P_{AA} shows that the subsequence is a *discord* in the T_A, which contradicts that the subsequence should present widely in Class A [35].

illustrate two issues of the baseline method with an example of the dataset ArrowHead in Figure 1.

1st issue: Discords as "shapelets". The first issue is that the largest difference indicator only considers the difference of distances between $\mathsf{dist}(\mathcal{S}, T_B)$ and $\mathsf{dist}(\mathcal{S}, T_A)$, *i.e.*, $||\mathsf{dist}(\mathcal{S}, T_B) - \mathsf{dist}(\mathcal{S}, T_A)||$ is the largest. There are two possible scenarios in the following and an example in Figure 6.

- One scenario is the larger $\mathsf{dist}(\mathcal{S}, T_B)$ (discord in T_B) and the smaller $\mathsf{dist}(\mathcal{S}, T_A)$ (motif in T_A). $\mathcal{S}$ appears in T_A but not in T_B, which shows $\mathcal{S}$ can be a shapelet of high quality.
- Another scenario is that $\mathcal{S}$ is a discord in both T_A and T_B, where $\mathsf{dist}(\mathcal{S}, T_A)$ and $\mathsf{dist}(\mathcal{S}, T_B)$ are both large. If the distances can be large, their difference can still be large. This does not necessarily indicate $\mathcal{S}$ can distinguish Class A from B but there is simply no subsequence in both classes that is similar to $\mathcal{S}$.

Assume another two scenarios are with the same difference between P_{AA} and P_{AB} value, and they are both the largest difference, which causes the shapelets classification error.

Example 1: One scenario is both the discord in Class B and Class A, the other is a discord in Class B and a motif in Class A. We formalize the scenario in the following.

$$\underbrace{||\mathsf{dist}(\mathcal{S}_1, T_B) - \mathsf{dist}(\mathcal{S}_1, T_A)||}_{\mathcal{S}_1 \text{ is a discord both in } T_B, T_A} = \underbrace{||\mathsf{dist}(\mathcal{S}_2, T_B) - \mathsf{dist}(\mathcal{S}_2, T_A)||}_{\mathcal{S}_2 \text{ is a discord in } T_B, \text{ but motif in } T_A}$$

Although the shapelet candidate in the second case (namely, $\mathcal{S}_2$) is better than $\mathcal{S}_1$ under the intuition of the shapelet, it is difficult for the MP baseline method to determine the final top-1 shapelet through Formula 4. □

2nd issue: Lack of shapelet diversity. The second issue is that the MP baseline method *concatenates* the time series of a class into one long time series (shown in Figure 1). The MP is built on the concatenated time series and the baseline method does not necessarily find shapelets that represent the whole class. When such shapelets are used for classification, they could not yield high accuracy. There are two specific issues about lack of diversity.

2.1: Classification ineffectiveness. This issue of the MP baseline method is that the discovered shapelets may only separate few time series instances. The baseline method concatenates all the time series instances into one long time series instance (*e.g.*, T_A, T_B), which shrinks the number of correctly classified instances for the final shapelets. From Formula 4, we can further investigate that T_A/T_B (Figure 1) is one long concatenated time series instance, which means the discovered shapelets are only aimed at the long time series instance T_A/T_B. However, the shapelets should classify all the time series instances from dataset D_C. For example, the top-1 shapelet can only classify some of the time series instances, even though the shapelet is the motif in Class A and the discord in Class B. We can realize that the accuracy of top-1 on the MP baseline method is much lower than those of 1NN-ED [9] and 1NN-DTW [9] in Table II.

TABLE II
THE ACCURACY OF THE DIFFERENT TOP-k SHAPELETS ON MP BASELINE, 1NN-ED, AND 1NN-DTW ON FOUR DATASETS FROM UCR ARCHIVE

The k in top-k	1	2	5	10	20	50	100	ED	DTW
ArrowHead	61.71	64	61.14	65.14	61.28	65.71	61.71	80	70.29
MoteStrain	69.88	77.47	77.08	78.59	77.02	77.39	78.19	87.79	83.47
ShapeletSim	52.23	55.56	54.44	58.33	60.56	57.77	56.11	53.89	65
ToeSegmentation1	66.66	67.1	70.18	68.86	71.49	72.36	71.93	67.98	77.19

2.2: Similar subsequences as shapelets. In practice, the shapelet diversity cannot be alleviated by simply selecting the top-k shapelets because there can be high similar shapelet candidates. It can be observed from Figure 1, Figure 3, and Figure 4. The top-5 shapelet indicators refer to similar subsequences in T_A, which limits its classification effectiveness.

To investigate the possibility of two issues, we study that the accuracies of the baseline's top-k shapelets by tuning k from 1 to 100. The results are reported in Table II. It can be observed that the baseline method of various ks has lower accuracies when compared to simple methods such as 1NN-ED [9] and 1NN-DTW [9]. On the contrary, the shapelets cannot distinguish on average 33% of time series from Class A and Class B even when k = 100. In all, two issues affect the quality of the discovered shapelets.

III. INSTANCE PROFILE FOR SHAPELETS (IPS)

In this section, we propose a novel shapelet discovery approach, called instance profile for shapelets (IPS), to address the two issues. We develop the essence of shapelet discovery in the following.

$$\arg\max_{\mathcal{S}} \sum_{\substack{T', T \in D \\ \mathsf{label}(T') \neq \mathsf{label}(T)}} ||\mathsf{dist}(\mathcal{S}, T') - \mathsf{dist}(\mathcal{S}, T)|| \tag{5}$$

We consider all the original time series instances from D in Formula 5 for supporting the robustness and improving the diversity of candidates, rather than one concatenated time series in Formula 4.

A. Overview of Instance Profile

We define instance profile (IP) formally and then present the shapelet candidate generation process with IP.

A time series dataset $D = \{D_1, D_2, \cdots, D_C, \cdots, D_{|\mathcal{C}|}\}$ is with M time series instances T_m of class label $C_m, m \in [1, M]$ of each instance length N. Specifically, time series instances from one class in D is $D_C = \{T_1, T_2, \cdots, T_{|D_C|}\}$, which can be concatenated into one long instance T_C, namely, $T_C = \mathsf{concatenate}(T_1, T_2, \cdots, T_{|D_C|})$.

Definition 8: **Instance profile**. Given a time series dataset D and a specific subsequence length L ($\leq N$), the *instance profile* is a data structure to annotate D, which consists of $IP(D_C, L)$, the instance profile of D_C, specifically,

$$IP(D_C, L) = IP(T_C, L) = (ip_1, ip_2, \cdots, ip_i, \cdots, ip_{|D_C|\cdot(N-L+1)}) \quad (6)$$

where ip_i is the minimum of the nearest neighbor distance of the subsequence $T_m^{i,i+L-1}$ among the other subsequences. Thus,

$$IP(D, L) = \big(IP(D_1, L), \cdots, IP(D_C, L), \cdots, IP(D_{|\mathcal{C}|}, L)\big) \quad (7)$$

Definition 9: **Subsequence Samples for** IP. The subsequences set S_C^L for $T_m^{i,i+L-1}$, to calculate its ip_i for the instance profile $IP(D_C, L)$, is generated from Q_S time series instances randomly selected of D_C, namely,

$$S_C^L = \bigcup_{m'=1}^{Q_S} T_{m'}^{i',i'+L-1} \quad (8)$$

where $m' \neq m, i' \in [0, N-L+1], Q_S \leq |D_C|$.
Thus,

$$ip_i = \min(\mathsf{dist}(T_m^{i,i+L-1}, T_{m'}^{i',i'+L-1})) \quad (9)$$

where $T_{m'}^{i',i'+L-1} \in S_C^L$. □

We further introduce the details of shapelet candidate generation in Algorithm 1. The empty set is initialized for each class to store the candidates (Line 2) and Q_N samples (Line 3) are generated for each class (a.k.a in a bagging way [5]). Each sample contains Q_S time series instances, which are randomly selected from D_C, to generate Q_C (Line 4). The instances are then concatenated into one long time series T_{Q_C} (Line 5). For the different lengths of the subsequences in ϕ (Line 6), we utilize the Definition 8 and Definition 9 to calculate instance profile then deduce the motifs and discords (Lines 7-8). Finally, the motif and discord are added into Φ_C (Lines 9-10).

Complexity analysis. The time complexity of Algorithm 1 is $O(|\mathcal{C}| \cdot Q_N \cdot |\phi| \cdot N^2)$, namely N^2, where N is the length of time series.

We select the motifs as final shapelet candidates for the 1st issue. The diversity of the candidates (2nd issue) is also significantly increased in Algorithm 1. However, it can computationally cost to discover high-quality shapelets from a large number of candidates. Let us use the ArrowHead dataset as an example. The class label, training set, and length are 3, 36, and 251, respectively. We set the sample number Q_N to 20, and use 5 different lengths of candidates. To discover top-k shapelets in ArrowHead requires $3 \times 20 \times 5 \times 251^2 \approx 1.89 \times 10^7$ combination trials using the naive method. We therefore propose the distribution-aware bloom filter (DABF) for pruning shapelet candidates and propose three utility functions for generating top-k shapelets efficiently.

Algorithm 1: Shapelet candidate generation with IP

Input: Dataset D, the class labels $\mathcal{C}$ of D, sample number Q_N, sample size Q_S, candidate lengths $\phi = \{l_1, \cdots, l_{|\phi|}\}$
Output: Candidate pool Φ

1 **foreach** $C \in \mathcal{C}$ **do**
2 initialize $\Phi_C = \emptyset$;
3 **for** $i = 0; i \leq Q_N; i++$ **do**
4 $Q_C = \mathsf{getsample}(D_C, Q_S)$; // random sampling
5 $T_{Q_C} = \mathsf{concatenate}(Q_C)$;
6 **for** $j = 0; j \leq |\phi|; j++$ **do**
7 $\Phi_C^{motif} = \min\big(\mathrm{IP}(T_{Q_C}, l_j)\big)$;
8 $\Phi_C^{discord} = \max\big(\mathrm{IP}(T_{Q_C}, l_j)\big)$;
9 $\Phi_C = \Phi_C \cup \{(\Phi_C^{motif}, \Phi_C^{discord})\}$;
  (Def.8 & Def.9)
10 $\Phi = \Phi \cup \Phi_C$;
11 **return** Φ

B. Distribution-aware bloom filter (DABF)

In this subsection, we first introduce the insight of distribution-aware bloom filter (DABF), which returns whether *a query is close to most elements in the set*. The bloom filter [4], [6] is a space-efficient index to test whether *a query is in the set*. The answer will be either "*possibly in the set*" or "*definitely not in the set*". The distance-sensitive bloom filter [15] is proposed to answer whether *a query is close to an element in the set*. Similar to the bloom filter [4], [6], the distance-sensitive bloom filter answer is either "*possibly close to an element*" or "*definitely not close*". However, the two previous filters are not specifically designed for our case to check whether a subsequence is *close to most elements* in Φ.

The intuitions of the solution are to determine two parameters, one for defining time series that are close to each other, and the other for quantifying the intuition of most elements. We then calculate the distance between the query and each element in the set. All the distances are compared with the parameters for determining whether the query is close to most elements in the set. The time complexity is however quadratic $O(N^2)$ for this naive method.

We next present the distribution-aware bloom filter (DABF) to solve the query in $O(N)$. A DABF is a data structure, which is used to test whether a query is *close to most elements* in a set. DABF of a dataset D is composed of DABF_C for each class, which consists of a set of locality-sensitive hashing (LSH) functions and the distance distribution of the time series subsequences, denoted as $\mathrm{DABF}_C = (\mathsf{LSH}_C, \mathsf{Distribution}_C)$. The DABF construction process is illustrated in Figure 7.

The LSH functions are utilized to hash the whole time series subsequences, after candidate generation, into the buckets (also regarded as clustering), which are ranked by the distance between the center of each bucket (cluster) and the original.

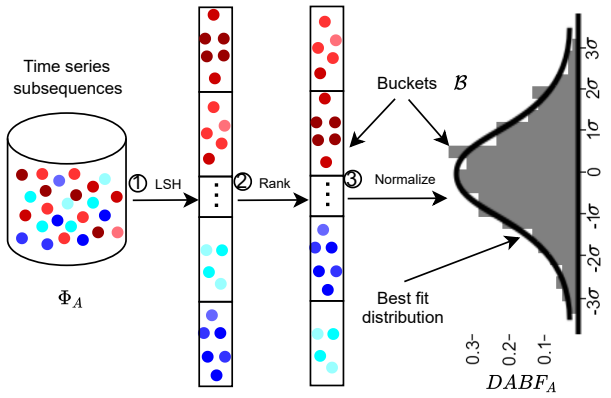


Fig. 7. The construction of distribution-aware bloom filter (DABF) for Class A. The locality-sensitive hashing (LSH) functions ① hash similar time series subsequences into the same bucket with a high probability. We then ② rank each bucket according to the distance between the bucket center and the original. The ③ z-normalization of the distance is calculated for the histogram to generate the best fit distribution.

TABLE III
THE BEST FIT DISTRIBUTION OF TEN UCR DATASETS ON DABF CONSTRUCTION UNDER NMSE

Dataset	Best fit distribution	NMSE	Dataset	Best fit distribution	NMSE
ArrowHead	Norm	0.073	GunPoint	Norm	0.208
BeelteFly	Norm	0.041	ItalyPower.	Norm	0.037
Coffee	Norm	0.085	Meat	Gamma	0.425
ECG200	Norm	0.019	Symbols	Norm	0.069
FordA	Norm	0.027	ToeSeg.1	Norm	0.179

Definition 10: **Locality-sensitive hashing (LSH) family** [14]. Let (X, M) be a metric space. Let the threshold and approximation factor be $r > 0$ and $c > 1$. An LSH family $\mathcal{H} = h : X \to U$ is called (r, cr, p_1, p_2)-sensitive, $\forall x, y \in X$,

- if $M(x, y) \leq r$, then $Pr_{h\sim\mathcal{H}}[h(x) = h(y)] \geq p_1$
- if $M(x, y) \geq cr$, then $Pr_{h\sim\mathcal{H}}[h(x) = h(y)] \leq p_2$

where the probability $p_1 > p_2$. □

We utilize the LSH scheme under $\mathcal{L}_2$ norm, based on p-stable distribution [7]. The experiments on other polular LSH schemes are presented in Table VII. The LSH is considered as a linear map preserving the distance similarity between two time series subsequences [29], stated in the Johnson–Lindenstrauss lemma [22]. The z-normalization of the distance is calculated for fitting the distribution efficiently (Formula 10).

$$\underset{\mathsf{Distribution}_C}{\arg\min} \sum_{B_i \in \mathcal{B}} ||\mathsf{Distribution}_C.\bar{i} - B_i||_2^2 \qquad (10)$$

where $\mathcal{B} = (B_1, B_2, \cdots, B_{|\mathcal{B}|}) = \mathsf{histogram}(\mathsf{LSH}_C(T_{a,b}))$, $T_{a,b} \in D_C$. The x-axis of $\mathsf{Distribution}_C$ can be divided into $|\mathcal{B}|$ segments. $\mathsf{Distribution}_C.\bar{i}$ is the average of the values in the ith segment of $\mathsf{Distribution}_C$. Table III shows the best fit distribution of 10 datasets on DABF construction under normalized mean square error (NMSE). It shows that 9 out of 10 datasets fit normal distribution and 7 datasets have an NMSE smaller than 10%. Hence, in practice, a distribution of the hashed time series subsequences do exists. The details of the DABF construction are in Algorithm 2.

Algorithm 2: Construction of DABF

Input: Candidate pool Φ
Output: Distribution-aware bloom filter DABF

```
1  initialize DABF = ∅ ;
2  foreach C ∈ 𝒞 do
3      initialize LSH_C ;                    // LSH for each class
4      foreach e ∈ Φ_C^motif or Φ_C^discord do
5          LSH_C.add(e) ;                    // Buckets inserting
6      ℬ = LSH_C.getbuckets() ;
7      rank ℬ by dist(B.center, 0), B ∈ ℬ ;
8      foreach B in ℬ do
9          B = (B − ℬ.μ)/ℬ.σ ;               // Normalize
10         Distribution_C = fit B into distribution ;  // (10)
11     DABF_C = (LSH_C, Distribution_C) ;
12     DABF = DABF.add(DABF_C) ;
13 return DABF
```

Algorithm 2 details the DABF construction. We initialize LSH functions (Line 3) for each class C (Line 2). For the subsequences in Φ_C, we add them into the different buckets using the LSH (Lines 4-5). The buckets are stored in $\mathcal{B}$ (Line 6). The distance between each bucket center and the original is computed for ranking the buckets (Line 7). The z-normalization of each bucket (Lines 8-9) is calculated for generating the histogram, then fitting the distribution further (Line 10). The DABF is constructed with the LSH and Distribution (Lines 11-12).

Complexity analysis. The time complexity of DABF construction is $O(|\mathcal{C}| \cdot |\Phi_C| \cdot N)$, where $|\mathcal{C}|$ is the label number, $|\Phi_C|$ is the number of candidates in class C, and N is the length of time series.

Since it is inefficient to discover final shapelets using the naive method from a large number of candidates generated by Algorithm 1, we exploit DABF to filter the candidates efficiently in Section III-C.

C. Shapelet candidate pruning with DABF

In this subsection, we introduce how to prune some candidates that do not fit the definition of shapelets with DABF. In our circumstance, we want to query whether a candidate is close to *most elements* in the set. Thus, it is important to define "*most elements*". Chebyshev's inequality states at least $(1 - \frac{1}{z^2})\%$ within z standard deviations (namely $z\sigma$) for a broad range of different probability distributions.

$$P(\mu - z\sigma \leq X \leq \mu + z\sigma) \geq 1 - \frac{1}{z^2} \qquad (11)$$

We apply the widely adopted 3σ-rule [28] from Chebyshev's inequality on the generated distribution to test whether the query is close to most elements, specifically, $P(|X - \mu| \leq 3\sigma) \geq 1 - \frac{1}{3^2} \approx 88.89\%$. The results are "*possibly close to most elements*" to prune the candidates and "*definitely not close to most elements*" to determine the candidates for final shapelets. The major steps of applying the DABF is illustrated with Figure 8.

We further elaborate on the details of shapelet candidate pruning with DABF in Algorithm 3. We initialize a 3σ-rule

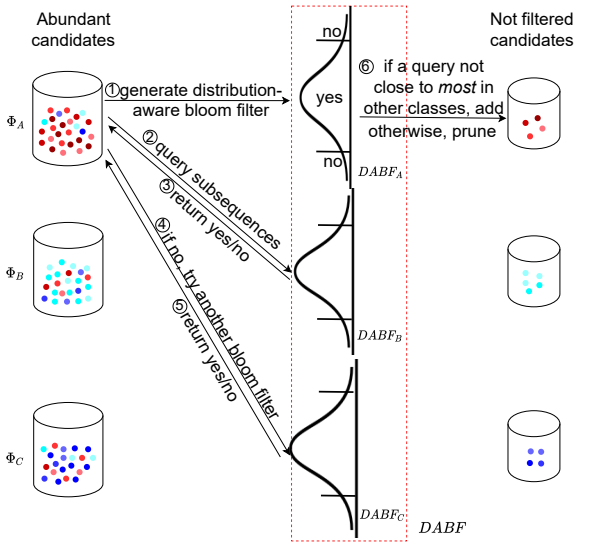


Fig. 8. Distribution-aware bloom filter (DABF) structure for pruning shapelet candidates. We generate a DABF for each class with Algorithm 2 (shown in ①) and take one candidate e from Φ_A as the query for the other two DABFs generated by Φ_B and Φ_C in ② and ④. Two operations are shown in ⑥. If the candidate can pass DABF, it may have a high discriminative power and can be maintained. Otherwise, the candidate is removed. Then, we continue the next candidate until all candidates in Φ_A are processed and carry out the same operation for candidates from Φ_B and Φ_C.

threshold θ (Line 1) for the DABF.Distribution to define "*most elements*". We then query the candidate e to test whether it is close to most elements from the DABF of other classes $\bar{C} \in \mathcal{C} \setminus \{C\}$ (Line 4). The candidate e is hashed to calculate the Euclidean distance $\mathsf{dist}(\mathsf{LSH}_{\bar{C}}(e), \mathbf{0})$, which is normalized by the mean μ and the standard deviation σ of $\mathsf{Distribution}_{\bar{C}}$. The normalized distance is utilized to test whether it is within the 3σ of the μ in the $\mathsf{Distribution}_{\bar{C}}$. From our experiments (Table VII), this calculation only leads to a slight drop in classification accuracies in practice. If the query result is "Yes", namely candidate within $\mu \pm 3\sigma$, it means that the motif candidate e is close to most elements at least in one of the other classes from $\bar{C}$, we then remove e (Line 6) and update the motif set Φ^{motif} (Line 7). The same operations are performed for the discord candidates (Lines 9-10). Otherwise, the DABFs return "No", which indicates this candidate is definitely not close to most elements in other classes, namely a good quality for the candidate to be the final shapelet.

Complexity analysis. The time complexity of Algorithm 3, shapelet candidate pruning with DABF, is $O(|\mathcal{C}| \cdot |\Phi_C| \cdot N)$.

It takes linear time $O(N)$ to query a candidate with the DABF, however, quadratic time $O(N^2)$ for the naive method. The DABF prunes candidates efficiently with high similarities, and does not reduce the diversity of the shapelet candidates, further the quality of candidates, finally increasing the performance.

D. Top-k shapelet selection

From the candidates not filtered by DABF, we score candidates and then select top-k, which are similar to the candidates

Algorithm 3: Shapelet candidate pruning with DABF

Input: Candidate pool Φ, DABF
Output: Pruned candidate Φ

1 initialize $\theta = 3\sigma$; // 3σ-rule threshold
2 **foreach** $C \in \mathcal{C}$ **do**
3 **foreach** $e \in \Phi_C$ *and* $\Phi_C \in \Phi$ **do**
4 **if** $\bigvee_{\bar{C} \in \mathcal{C} \setminus \{C\}} \left(\mathrm{DABF}_{\bar{C}}.\mathsf{Distribution}_{\bar{C}}.\mathsf{query}(e, \theta, \leq)\right)$ **then**
5 **if** $e \in \Phi_C^{motif}$ **then**
6 $\Phi_C^{motif}.\mathsf{remove}(e)$;
7 $\Phi_C.\mathsf{update}(\Phi_C^{motif})$;
8 **else**
9 $\Phi_C^{discord}.\mathsf{remove}(e)$;
10 $\Phi_C.\mathsf{update}(\Phi_C^{discord})$;
11 **return** Φ

of its class (small intra class distance) and dissimilar to the candidates of other classes (large inter class distance). In addition to the candidates' perspective, we consider the time series instances from the intra class to further avoid the scenario in Example 1.

We propose three utility functions for scoring the candidates from the intra class, inter classes, and time series instances, respectively. We explain the details of selecting shapelets for class C. The motif candidates in class C are scored by intra-class utility (Definition 11), which means the total distance between one of the motif candidates and other candidates. When the total distance of the candidate is small, it indicates that the motif candidate can represent other motif candidates in class C (one shapelets' attribute of *widely existing in intra class*). The smaller the total distance among the motif candidate is, the more likely the candidate is a final shapelet.

Definition 11: **Intra-class utility**. The intra-class utility is the distance between the *motif* candidate Can_i and other *motif* candidates $\sum_{j=1}^{|\Phi_C|} Can_j$ is denoted as,

$$U_{intra}(Can_i) = \frac{1}{1 + e^{-\sum_{j=1}^{|\Phi_C|} \mathsf{dist}(Can_i, Can_j)}} \tag{12}$$

where both Can_i and Can_j are from intra class C. □

We then employ the motifs/discords from the inter classes $\{\mathcal{C} \setminus C\}$ to score motif candidates (from class C) by inter-class utility (Definition 12). The large value computed by Formula 13 implies that the motif candidate (from class C) has a large distance with motifs/discords (from $\{\mathcal{C} \setminus C\}$), which signifies another shapelets' attribute, *far away from the other classes*.

Definition 12: **Inter-class utility**. The inter-class utility is the distance between the *motif* candidate Can_i and other *motif/discord* candidates Can_j is denoted as,

$$U_{inter}(Can_i) = \frac{1}{1 + e^{-\sum_{j=1}^{|\Phi_{\{\mathcal{C} \setminus C\}}|} \mathsf{dist}(Can_i, Can_j)}} \tag{13}$$

where Can_i from class C and Can_j from $\{\mathcal{C} \setminus C\}$. □

We next incorporate the distance between motif candidates and the raw time series instances also from class C into the score. Intra-instance utility (Definition 13) is designed for the characteristics of shapelets that *the time series instances in the same class should be closer to the shapelet*.

Definition 13: **Intra-instance utility**. The utility is the distance between the *motif* candidate Can_i and time series T_j is denoted as,

$$U_{D_C}(Can_i) = \frac{1}{1 + e^{-\sum_{j=1}^{|D_C|} \mathsf{dist}(Can_i, T_j)}} \tag{14}$$

where both Can_i and T_j are from class C. □

Brute force method. To calculate each utility and sum the three utilities for each motif candidate, then sort the scores for selecting the final top-k candidates, is however time-consuming $O(|\mathcal{C}| \cdot |\Phi_C| \cdot N^2)$.

E. Optimization of utility computation

We propose the following two techniques, namely distribution transformation (DT) and computation reuse (CR), for computing the final candidates efficiently.

1) Distribution transformation (DT): We transform the raw subsequences/candidates distance calculation, namely $\mathsf{dist}(Can_i, Can_j)$, into a lower bound for Euclidean space [7] through DABF.

We employ the intra-class utility (Definition 11) to clarify it in detail. The hash of Can_i and Can_j is calculated in the DABF.Distribution. The $\mathsf{dist}(Can_i, Can_j)$ is then transformed into the distance in the Distribution space.

$$\begin{aligned} \mathsf{dist}(Can_i, Can_j) &\geq \frac{||\mathsf{LSH}(Can_i) - \mathsf{LSH}(Can_j)||}{(1+\epsilon)} \\ &> \frac{||\mathsf{LSH}(Can_i) - \mathsf{LSH}(Can_j)||}{2} \\ &= |B_i - B_j| \end{aligned} \tag{15}$$

where $0 < \epsilon < 1$ and we use B_i to simplify the formula.

The original intra-class utility is reformulated as follows,

$$U_{intra}(Can_i) > \tilde{U}_{intra}(Can_i) = \frac{1}{1 + e^{-\sum_{j=1}^{|\Phi_C|} |B_i - B_j|}} \tag{16}$$

The similar transformation can be employed for the inter-class utility (Formula 13) and intra-instance utility (Formula 14).

Analysis. If two candidates from the same class are hashed into the same bucket. Then, the final score, namely the sum of U_{intra}, U_{inter}, and U_{D_C} of candidates, is the same. The error of the distance between two candidates is less than the width of the bucket. As a result, the time complexity of the calculation is reduced from $O(N^2)$ to $O(N)$.

Algorithm 4: Top-k shapelet generation with optimization

Input: Dataset D, pruned candidate Φ, DABF, shapelet number k
Output: Final shapelets $\mathcal{S}$

```
1  for C ∈ 𝒞 do
2      initialize 𝒮_C = ∅ ;
3      initialize Q = ∅ ;                          // Priority queue
4      for e ∈ Φ_C do
5          initialize u = 0 ;
6          u = Ũ_intra(e) − Ũ_inter(e) + Ũ_{D_C}(e) ; // DT&CR
7          Q.add(u) ;
8      for i = 0; i < k; i++ do
9          j = Q.poll() ;
10         𝒮_C = 𝒮_C ⋃ {Φ_C^motif[j]} ;
11     𝒮 = 𝒮 ⋃ 𝒮_C ;
12 return 𝒮
```

2) Computation reuse (CR): One reason for inefficient computation is the numerous repeated utility calculation. For example, when we calculate the intra-class utility (Formula 12) for one candidate, the distance between candidate and other candidates will be computed again for the intra-class utility of other candidates. Thus, we reuse the computation results for improving efficiency. We calculate the distances between every two candidates, then combine the distances for each candidate's utility, which reduces the computation time in half. We also observe the similar repeated computation in inter-class utility (Formula 13).

Putting this together. Algorithm 4 is the pseudo-code for determining top-k shapelets. First, we initialize the set of shapelets $\mathcal{S}$ (Line 2). For each class C, we also initialize the priority queue, Q, to store the utilities (Line 3). For each candidate in Φ_C, the three utilities are calculated and added into Q (Lines 4-7). We then poll Q for generating the final top-k shapelets (Lines 8-11).

Complexity analysis. The time complexity of Algorithm 4 is $O(|\mathcal{C}| \cdot |\Phi_C| \cdot N \cdot \log(|\Phi_C|))$, close to $O(|\Phi_C| \cdot N)$. The full complexity is $O\big(|\mathcal{C}| \cdot [|\Phi_C| \cdot N \cdot \log(|\Phi_C|) + k]\big)$.

Remarks. The final shapelets are then employed to transform the original time series [3], [24], [26] for learning a classification model. In this paper, we adopt SVM with a linear kernel for the classification.

IV. Experimental Evaluation

In this section, we report the experiments of IPS with 12 related methods on the widely-used UCR datasets [9]. The setup used in the previous works [2], [23] have been followed. We report the overall efficiency and accuracy of IPS on the 46 datasets [2]. Due to the space restriction, we highlight the results of some datasets and interpret the discovered shapelets with one case study.

[2]These datasets are reported in the experiments of other papers [2], [23]. The full results of UCR datasets [9] can be found in a technical report [25], which is publicly available at the project website https://www.comp.hkbu.edu.hk/~csgzli/ips/.

A. Experimental Settings

Environment. We have implemented the proposed method in PYTHON. All the experiments were conducted on a machine with a Xeon E5-2630 v4 @ 2.2GHz (2S/10C) / 256GB RAM / 128GB SWAP, running on CentOS 7.6 (64-bit). The storage is a HDD, and its capacity is 1.8TB.

Compared methods. The 12 compared methods are, namely, Rotation Forest [2] and 1NN-DTW [32], and many other shapelet-based methods, including ST [26], LTS [16], Fast shapelets [30], SD [17], ELIS [11], COTE (a meta-ensemble method combined with 35 different classifiers [3]), ResNet (one of the best deep learning methods [34]), BSPCOVER(the current state-of-the-art efficient method [23]), COTE-IPS (COTE augmented by IPS) and BASE (the MP baseline method [37]). Due to space limitations, we omit the details of each method in this paper. Interested readers may refer to the original papers for details.

Parameter setting. Some parameters adopted in the experiments are listed as follows. For fairness, we set the shapelet number to 5 in both BASE and IPS. The lengths of shapelet candidates are given as a ratio of the subsequence length to the length of the original time series. It ranges among these values $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. (*e.g.*, 0.2 means that the subsequence's length is 20% of the original time series' length.) The sample number Q_N and sample size Q_S are selected from $\{10, 20, 50, 100\}$ and $\{2, 3, 4, 5, 10\}$, respectively.

B. Experiments on Efficiency

From the previous work, we observe that COTE and ST are clearly slower than LTS [2], and the efficiency of ELIS is about two orders of magnitude faster than that of LTS [11], and BSPCOVER is 70x faster than ELIS [23]. Thus, we compare the efficiency of BASE and BSPCOVER with that of IPS.

Comparison with BASE and BSPCOVER. The total running time of BASE, BSPCOVER, and IPS, and the improvement of BASE over IPS and that of IPS over BSPCOVER, reported in Table IV (The unit of the numbers is seconds).

IPS is consistently faster than BSPCOVER on these 46 datasets. It is on average about $25\times$ faster, in terms of the total running time. The total running time of BASE is only slightly faster than that of IPS ($1.22\times$ on average), however, as we shall see its accuracy is lower in 41 out of 46 datasets (Table VI). Hence, IPS discovers high-quality shapelets to enhance the final accuracy with an efficiency similar to BASE.

Efficiency performance breakdown of the three main steps of IPS. Table V shows the running time of the candidate generation, pruning with/without DABF, and top-k selection with/without optimization techniques. We can observe that the time of candidate generation takes 4%-20% of the overall time. The DABF and DT & CR optimization techniques save at least 50% running time when compared with the naive method for pruning and using raw time series data for discovering the top-k shapelet.

Efficiency by varying the shapelet number k. Figure 9 shows the total running time (line chart) and the accuracy (bar chart) with the increase of the shapelet number k on two datasets (BeetleFly and TwoLeadECG), and three methods, namely BASE, IPS, and BSPCOVER.

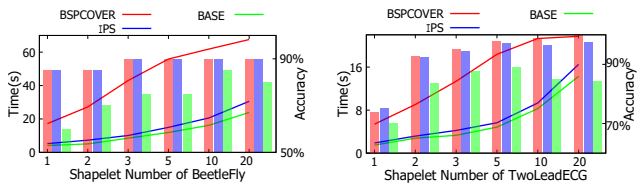
The accuracy of BASE on both two datasets is significantly lower than that of IPS. The accuracy of IPS is similar to BSPCOVER. The number of shapelets contributes to the accuracy at the beginning and then the accuracies stabilize. For the efficiency on BeetleFly, IPS and BASE maintain an approximately linear relationship with k. The runtime of BSPCOVER is larger than the two methods, whereas the differences between the runtimes of BASE and IPS are not significant. Regarding TwoLeadECG, a similar trend can be observed

TABLE IV
EFFICIENCY OF IPS AND RELATED METHODS ON UCR ARCHIVE (SECOND AS UNIT) AND THE SPEEDUP

Dataset	BASE(s)	BSPCOVER(s)	IPS(s)	Speedup BASE vs IPS	Speedup IPS vs BSPCOVER
ArrowHead	7.65	55.57	10.57	1.38	5.26
Beef	10.56	131.17	15.42	1.46	8.51
BeetleFly	16.19	42.92	16.46	1.02	2.61
CBF	4.53	16.43	4.85	1.07	3.39
ChlorineConcentration	29.17	173.86	29.66	1.02	5.86
Coffee	5.15	10.96	6.33	1.23	1.73
Computers	103.63	1049.52	104.99	1.01	10.00
CricketZ	641.89	20993.38	756.90	1.18	27.74
DiatomSizeReduction	11.87	30.04	13.04	1.10	2.30
DistalPhalanxOutlineCorrect	12.67	52.39	16.76	1.32	3.13
Earthquakes	178.06	2957.36	179.97	1.01	16.43
ECG200	9.17	48.34	13.49	1.47	3.58
ECG5000	30.01	600.37	38.35	1.28	15.65
ECGFiveDays	1.06	1.38	1.11	1.04	1.25
ElectricDevices	202.86	20851.50	251.53	1.24	82.90
FaceAll	108.63	1541.80	122.38	1.13	12.60
FaceFour	9.59	32.67	9.83	1.02	3.33
FacesUCR	5.19	1265.71	7.04	1.36	179.80
FordA	236.45	37481.21	255.09	1.08	146.94
GunPoint	2.28	8.97	3.05	1.34	2.94
Ham	11.49	126.13	21.91	1.91	5.76
HandOutlines	607.26	4340.86	623.87	1.03	6.96
Haptics	504.48	11523.26	590.07	1.17	19.53
InlineSkate	993.56	15060.30	989.82	1.00	15.22
InsectWingbeatSound	169.25	646.49	172.25	1.02	3.75
ItalyPowerDemand	0.45	2.91	0.67	1.49	4.34
LargeKitchenAppliances	412.52	13974.8	488.02	1.18	28.64
Mallat	135.05	2896.15	159.36	1.18	18.17
Meat	8.85	44.02	9.37	1.06	4.70
NonInvasiveFatalECGThorax1	15385.63	40125.42	15806.39	1.03	2.54
OSULeaf	99.48	6753.46	110.42	1.11	61.16
Phoneme	3586.33	45767.83	3812.99	1.06	12.00
RefrigerationDevices	1258.35	8871.13	1563.59	1.24	5.67
ShapeletSim	30.08	455.23	39.26	1.31	11.60
SonyAIBORobotSurface1	2.39	4.19	2.89	1.21	1.45
SonyAIBORobotSurface2	1.65	3.78	2.59	1.57	1.46
Strawberry	15.64	235.17	18.87	1.21	12.46
Symbols	10.11	90.43	15.85	1.57	5.70
SyntheticControl	5.36	249.29	6.01	1.12	41.47
ToeSegmentation1	1.62	19.91	2.71	1.67	7.36
TwoLeadECG	8.27	20.32	8.98	1.09	2.26
TwoPatterns	149.16	17891.24	152.13	1.02	117.60
UWaveGestureLibraryY	956.34	193667.30	998.26	1.04	194.00
Wafer	50.49	825.96	56.88	1.13	14.52
WormsTwoClass	305.49	1124.08	321.57	1.05	3.50
Yoga	207.58	10593.18	227.35	1.10	46.59
Average				**1.20**	**25.74**

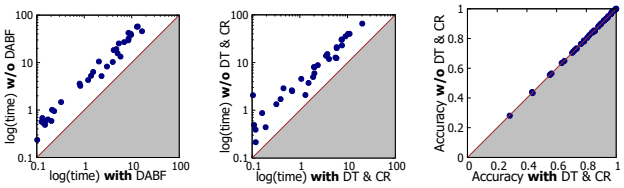
TABLE V
THE EFFICIENCY OF THREE PARTS ON FOUR DATASETS FROM UCR ARCHIVE (SECOND AS UNIT)

Datasets	Candidate generation	Pruning without DABF	Pruning with DABF	Without DT+CR	With DT+CR
ArrowHead	0.71	0.57	0.13	2.07	0.11
Computers	3.28	25.32	5.35	13.82	3.55
ShapeletSim	1.23	8.25	3.01	3.74	1.52
UWaveG.Y	12.76	209.47	22.68	39.97	10.26



(a) BeetleFly (b) TwoLeadECG

Fig. 9. Efficiency vs Top-k (shapelet number) of BASE, IPS, BSPCOVER methods on BeetleFly and TwoLeadECG, respectively



(a) (b) (c)

Fig. 10. (a) Efficiency with and without DABF for candidates pruning on the UCR datasets. (b) Efficiency and (c) accuracy with and without the optimization techniques (LT & CR) on UCR datasets

when $k \leq 10$ among the three methods. Nevertheless, the total time stabilizes when $k > 10$ in BSPCOVER and still grows rapidly in BASE, IPS. All the shapelet candidates have been determined when $k = 10$ in BSPCOVER [23].

Efficiency evaluation with and without DABF. We compare DABF with the naive methods on the efficiency of candidates pruning. Figure 10(a) reports the running time of DABF construction and pruning process and that of the naive method in log space. We can observe that all the points (datasets) are in the upper triangle, which indicates that it takes more time for pruning without DABF. DABF speeds up the naive method from 2 to 10 times.

Efficiency and accuracy with and without the optimization techniques (DT & CR). The DT and CR optimization techniques in Section III-E are compared from the efficiency and accuracy. Figure 10(b) illustrates the running time with and without DT & CR techniques for the top-k shapelet selection. We can observe that all the points (datasets) are in the upper triangle. The optimization techniques save 50% to 90% of the shapelet discovery time. Meanwhile, the accuracy with and without the two optimization techniques is similarly shown in Figure 10(c). From the results, we can see that IPS has taken advantages of the efficiency of DABF for shapelet discovery.

C. Experiments on Accuracy

The experiment accuracy results of 10 methods (with the exception of COTE-IPS and BASE) are all taken from the papers [2], [12], [23]. The results of IPS, COTE-IPS, and BASE are the mean values of 5 runs and the standard deviations of all the datasets are less than 0.01. Due to space limitations, we provide some highlights of accuracy results with some selected datasets of different types, *e.g.*, Image, Sensor, Simulated, and Motion. The full results of UCR datasets [9] can be found in a technical report [25].

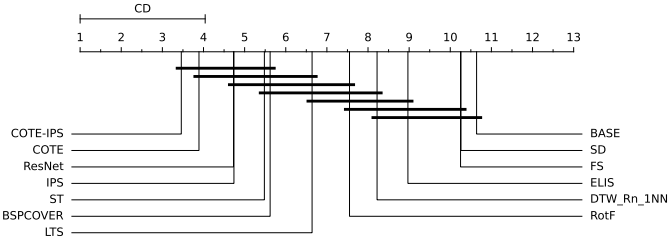


Fig. 11. Critical difference diagram of the pairwise statistical comparison of 13 methods on the UCR Archive. A thick horizontal line groups a set of classifiers that are not significantly different.

Comparison with other methods. The overall accuracy results for 46 datasets are presented in Table VI. We can observe that the overall accuracy of IPS is ranked the $4th$ among all methods, where the $1st$ is not surprisingly the ensemble method COTE-IPS. Furthermore, IPS performs the best in 9 datasets (ranked $3rd$), only lower than the ensemble method, COTE-IPS and COTE, but much higher than DTW_Rn_1NN, LTS, FS, SD, ELIS, and BASE. IPS achieves much higher accuracies in some datasets, such as RefrigerationDevices and SonyAIBORobotSurface1. The 1-to-1-Losses datasets on accuracy are marginally lower than ST (*e.g.*, CricketZ and FaceAll) and BSPCOVER (*e.g.*, Haptics and InlineSkate). The accuracy of BASE is much lower than that of IPS.

Friedman test and Wilcoxon test. The Friedman test [10], a non-parametric statistical test, and Wilcoxon-signed rank test with Holm's α (5%) [19] are taken for all methods.

The Friedman test is to detect the differences among 46 datasets across 13 methods. The statistical significance p-value is 0.00, which is smaller than $\alpha = 0.05$. Thus, we reject the null hypothesis and a significant difference among all methods.

We then conduct the post-hoc analysis among all the methods. The results are visualized by a critical difference diagram in Figure 11. We observe that IPS significantly outperforms other approaches except for COTE, COTE-IPS, ResNet, ST, and BSPCOVER. The cd-diagram in Figure 11 further validates that the accuracy of BASE is poor.

Accuracy by varying three LSH functions, Hamming, Cosine, and $\mathcal{L}_2$. We investigate the accuracy performance on a different kinds of LSH functions, such as Hamming, consine, and $\mathcal{L}_2$ in table VII. We can observe that the accuracy of $\mathcal{L}_2$ is much better than that of other two functions. The accuracy of Hamming is the lowest, which shows that the Hamming distance is not suitable for time series. The accuracy of consine function is slightly lower than that of $\mathcal{L}_2$, which indicates that it can catch the distance among time series.

Accuracy by varying the shapelet numbers k. We investigate the performance on different numbers of shapelets from $\{1, 2, 5, 10, 20\}$. The impact of different shapelet numbers on the final accuracy of IPS on four datasets, Arrowhead, MoteStrain, ShapeltSim, and ToeSegmentation1.

TABLE VI
ACCURACY (%) OF IPS AND RELATED METHODS ON UCR ARCHIVE

Dataset	RotF	DTW_Rn_1NN	ST	LTS	FS	SD	ELIS	BSPCOVER	ResNet	COTE	COTE-IPS	BASE	IPS
ArrowHead	73.71	80	73.71	84.57	59.43	65.7	81.43	80.57	84.5	81.14	84	61.14	**85.14**
Beef	86.67	66.67	**90**	86.67	56.67	50.7	63.33	73.33	75.3	86.67	**90**	50	73.33
BeetleFly	**90**	65	**90**	80	70	75	85	**90**	85	80	**90**	75	**90**
CBF	92.89	99.44	97.44	99.11	94	97.5	90.44	99.67	99.5	99.56	**99.78**	68	**99.78**
ChlorineConcentration	**84.74**	65	69.97	59.24	54.64	55.3	27.39	61.22	84.4	72.71	70.5	54.66	63.41
Coffee	**100**	**100**	96.43	**100**	92.86	96.1	96.43	**100**	**100**	**100**	**100**	95.14	**100**
Computers	70	62.4	73.6	58.4	50	58.8	50	67.2	**81.5**	74	74	66.8	74
CricketZ	65.64	73.59	78.72	74.1	46.41	67.3	78.95	74.1	81.2	**81.54**	**81.54**	37.44	78.46
DiatomSizeReduction	87.25	93.46	92.48	**98.04**	86.6	89.6	89.86	87.25	30.1	92.81	92.81	89.2	88.89
DistalPhalanxOutlineCorrect	75.72	72.46	77.54	77.9	75	71.7	57.83	83.17	71.7	76.09	80.17	78.83	**83.67**
Earthquakes	74.82	72.66	74.1	74.1	70.5	63.6	77.64	81.68	71.2	74.82	78.99	**81.99**	**81.99**
ECG200	85	88	83	88	81	81.8	80	**92**	87.4	88	88	88	88
ECG5000	94.58	92.51	94.38	93.22	92.27	92.4	72.69	94.44	93.4	**94.6**	94.44	92.34	94.44
ECGFiveDays	90.82	79.67	98.37	**100**	99.77	95.3	95.45	**100**	97.5	99.88	99.88	77.82	99.88
ElectricDevices	**78.58**	63.08	74.7	58.75	57.9	59.3	8.65	24.24	72.9	71.33	70.6	53.99	55.47
FaceAll	91.12	80.77	77.87	74.85	62.6	71.4	75.56	76.33	83.9	**91.78**	85.6	70.18	76.36
FaceFour	81.82	89.77	85.23	**96.59**	90.91	82	95.46	**96.59**	95.5	89.77	91.58	81.82	92.78
FacesUCR	80.29	90.78	90.59	93.9	70.59	84.7	63.63	78.29	95.5	**94.24**	93.9	67.61	80.58
FordA	84.47	66.52	**97.12**	95.68	78.71	77.6	67.6	96.31	92	95.68	94.12	63.32	84.78
GunPoint	92	91.33	**100**	**100**	94.67	93.1	97.57	**100**	99.1	**100**	**100**	82.67	**100**
Ham	71.43	60	68.57	66.67	64.76	61.9	63.81	**76.19**	75.7	64.76	69.68	68.57	72.38
HandOutlines	91.08	87.84	**93.24**	48.11	81.08	79.9	/	86.7	91.1	91.89	90.62	73.8	89.9
Haptics	43.83	41.56	**52.27**	46.75	39.29	35.6	41.56	45.13	51.9	**52.27**	**52.27**	30.19	43.51
InlineSkate	37.09	38.73	37.27	43.82	18.91	38.5	35.46	38.73	37.3	**49.45**	48.75	21.27	43.82
InsectWingbeatSound	63.64	57.37	62.68	60.61	48.94	44.1	59.55	57.42	50.7	**65.25**	63.55	17.63	56.52
ItalyPowerDemand	**97.28**	95.53	94.75	96.02	91.74	92	96.57	96.5	96.3	96.11	96.11	92.63	96.6
LargeKitchenAppliances	60.8	79.47	85.87	70.13	56	57.1	33.33	86.13	**90**	84.53	84.53	57.6	85.34
Mallat	94.93	91.43	96.42	95.01	**97.61**	92.6	81.58	76.8	97.2	95.39	95.39	90.54	94.69
Meat	96.67	93.33	85	73.33	83.33	93.3	55	75	**96.8**	91.67	92.88	93.33	93.33
NonInvasiveFatalECGThorax1	90.53	82.9	**94.96**	25.9	71.04	81.4	/	91.47	94.5	93.13	93.13	56.74	92.06
OSULeaf	57.02	59.92	96.69	77.69	67.77	56.6	76.45	83.88	**97.9**	96.69	95.45	57.44	71.49
Phoneme	12.97	22.68	32.07	21.84	17.35	15.8	15.19	20.73	33.4	**34.92**	33.58	18.41	28.43
RefrigerationDevices	56.53	44	58.13	51.47	33.33	46.1	40	54.67	52.5	54.67	58.67	49.87	**78.33**
ShapeletSim	41.11	69.44	95.56	95	**100**	67.2	**100**	84.44	77.9	96.11	96.67	54.44	84.33
SonyAIBORobotSurface1	80.87	69.55	84.36	81.03	68.55	85	87.85	88.35	95.8	84.53	92.4	87.35	**98.5**
SonyAIBORobotSurface2	80.8	85.94	93.39	87.51	79.01	78	93.17	93.49	**97.8**	95.17	93.84	82.78	91.71
Strawberry	97.3	94.59	96.22	91.08	90.27	88.4	83.85	94.29	**98.1**	95.14	96.9	87.6	96.72
Symbols	79.3	93.77	88.24	93.17	93.37	90.1	78.29	93.37	90.6	**96.38**	**96.38**	69.45	94.1
SyntheticControl	97.33	98.33	98.33	99.67	91	98.3	99.33	99.67	99.8	**100**	**100**	94.67	99.67
ToeSegmentation1	53.07	75	96.49	93.42	95.61	88.2	**98.24**	96.49	96.3	97.37	97.37	70.18	96.49
TwoLeadECG	97.01	86.83	99.74	99.65	92.45	86.7	99.82	99.65	**100**	99.3	99.3	88.85	97.1
TwoPatterns	92.8	99.85	95.5	99.33	90.83	98.1	99.75	99.8	**100**	**100**	**100**	91.5	99.05
UWaveGestureLibraryY	71.44	70.18	73.03	70.3	59.58	67.1	69.32	64.01	67	**75.85**	**75.85**	53.81	65.21
Wafer	99.45	99.59	**100**	99.61	99.68	99.3	99.43	99.81	99.9	99.98	99.98	96.24	99.51
WormsTwoClass	68.83	58.44	**83.12**	72.73	72.73	64.1	71.82	74.59	74.7	80.52	80.52	42.54	73.48
Yoga	82.43	84.3	81.77	83.43	69.5	62.5	83.9	**88.2**	87	87.67	87.67	70.53	85.73
Total best acc	5	1	9	5	2	0	2	8	9	14	11	1	9
IPS 1-to-1 Wins	30	34	20	26	42	42	35	23	18	13	10	41	-
IPS 1-to-1 Draws	2	3	3	5	0	0	0	7	1	4	8	2	-
IPS 1-to-1 Losses	14	9	23	15	4	4	11	16	27	29	28	3	-

TABLE VII
JUSTIFICATION OF THREE LSH FUNCTIONS, HAMMING, COSINE, $\mathcal{L}_2$ ON ACCURACY (%) OF TEN UCR DATASETS

Dataset	LSH	Accuracy	Dataset	LSH	Accuracy
ArrowHead	Hamming	78.22	GunPoint	Hamming	91.33
	Cosine	84.31		Cosine	97.28
	$\mathcal{L}_2$	**85.14**		$\mathcal{L}_2$	**100**
BeelteFly	Hamming	80	ItalyPower.	Hamming	92.8
	Cosine	85		Cosine	94.7
	$\mathcal{L}_2$	**90**		$\mathcal{L}_2$	**96.6**
Coffee	Hamming	95.69	Meat	Hamming	83.33
	Cosine	96.1		Cosine	**93.33**
	$\mathcal{L}_2$	**100**		$\mathcal{L}_2$	**93.33**
ECG200	Hamming	80	Symbols	Hamming	70.82
	Cosine	**88**		Cosine	89.07
	$\mathcal{L}_2$	**88**		$\mathcal{L}_2$	**94.1**
FordA	Hamming	79.72	ToeSeg.1	Hamming	76.54
	Cosine	80.82		Cosine	82.91
	$\mathcal{L}_2$	**84.78**		$\mathcal{L}_2$	**96.49**

Figure 12 shows the accuracy by varying the shapelet numbers. Four different datasets show different trends, which reveal the appropriate shapelet number of the dataset. For example, the accuracy stabilizes in ToeSegmentation1 through all the shapelet numbers and peaks at 5. Thus, the shapelet number of ToeSegmentation1 is 5. The accuracy rises rapidly with the increase of the shapelet number from 1 to 5 on MoteStrain and then remains stable. Therefore, its shapelet number is set to 5.

D. Experiments on Interpretability

We further investigate a strength of shapelets, the interpretability. We compare the shapelet discovered by IPS and BSPCOVER in the following on ItalyPowerDemand.

We use ItalyPowerDemand[3] as it can be concisely explained

[3]http://www.timeseriesclassification.com/description.php?Dataset=ItalyPowerDemand

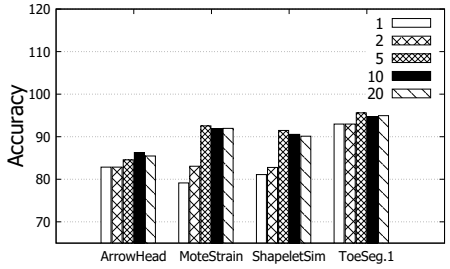
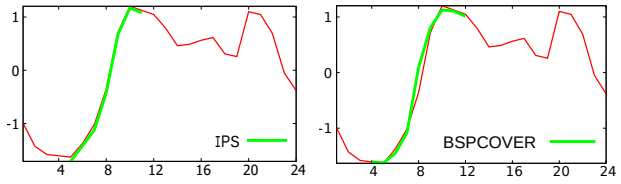
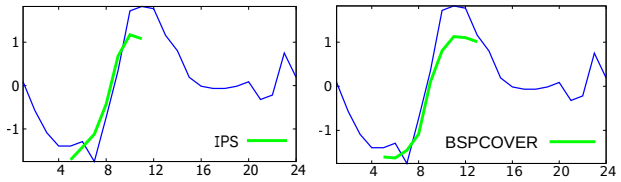


Fig. 12. Accuracy by varying 5 numbers of shapelets on four datasets



(a) Instance from Class 1



(b) Instance from Class 2

Fig. 13. Different shapelets (green) from IPS and BSPCOVER of ItalyPowerDemand highlighting the morning heating demand difference of summer (red) and winter (blue) months

to non-domain experts. As the name indicates, ItalyPowerDemand describes the power demand classified by summer and winter in Italy.

From the two left subfigures of Figure 13, the shapelet $\mathcal{S}$ (green color) demonstrates the difference between two classes, which are similar to the shapelet learned by BSPCOVER (two right subfigures in Figure 13). Specifically, both shapelets can indicate the power demand in winter (Class 2) is higher corresponding to the heating. The difference between the shapelets discovered by IPS and BSPCOVER is minor but IPS is 4 times faster than BSPCOVER.

V. Related Work

Readers who are interested in a general overview of the methods for time series classification may refer to an excellent review paper [2]. In this section, we focus on the matrix profile and shapelet-based methods.

A. Matrix Profile (MP) for motif/discord mining

Recently, MP methods (*e.g.*, [1], [8], [36]), applied to time series motif discovery and anomaly detection, are both efficient and effective. MP [37] is an annotation series, where each element is the nearest neighbor distance of the subsequences. GPU-STOMP [38] is combined with a high-performance GPU to improve the scalability of motif/discord discovery significantly. Alaee et al. [1] present the first efficient, scalable and exact method to find motifs under DTW. This method automatically performs the best trade-off of time-to-compute versus tightness-of-lower-bounds for a novel hierarchy of lower bounds. He et al. [18] demonstrated the essence of MP, 1-NN based nonparametric density estimation of subsequences. An ensemble framework, called neighbor profile (NP) [18], was developed to robustly estimate the subsequence density with kNN in a bagging way. The method for discovering shapelets from NP is not presented. We follow the description of MP [37] to implement the MP baseline method (BASE).

B. Shapelet-based methods for TSC

Shapelets were originally introduced for time series mining with the interpretability in [35]. There have been follow-up studies on shapelets, including logical shapelets [27], shapelet transformation [26], learning shapelets [16], efficient learning shapelets [20], and efficient shapelet discovery [23]. Grabocka et al. provide a fast shapelet discovery method [17] with a distance-based clustering technique for similar shapelets pruning and an incremental NN classifier. Rakthanmanon et al. propose fast shapelets [30], which utilizes a random masking strategy on SAX words of raw subsequences to efficiently discover shapelets. The time complexity is significantly reduced from $O(M^2N^3)$ to $O(MN^2)$ when compared to [35], but the accuracy is often lower than in some recent work [2].

There have been recent studies on employing neural networks to solve the time series classification problem (*e.g.*, an excellent survey of deep learning methods for TSC [12]). An extensive empirical study [12] shows that ResNet [34] is one of best the deep learning methods for TSC. As motivated, this paper undertakes the matrix profile-based shapelet approach by taking the advantage of the efficiency of matrix profile, and the accuracy and interpretability of shapelets.

VI. Conclusion

This paper has presented a study of an efficient shapelet discovery for TSC. We start with an efficient primitive for time series analysis called matrix profile (MP). We analyze the inherent two issues of the MP baseline method [37]. To address the issues, we propose IPS, a novel efficient instance profile-based shapelet approach for TSC. We obtain the motifs with the instance profile (IP) from time series instances in a sample way, which allows a large quantity of diverse shapelet candidates. The distribution-aware bloom filter (DABF) structure is proposed for pruning a large number of similar candidates. We then propose two optimization techniques (namely, DT & CR) with our DABF to efficiently score candidates for determining the final shapelets. The experimental results demonstrate that IPS has successfully taken the superior efficiency of MP and proposes highly effective shapelets for TSC. We employ the ItalyPowerDemand dataset to illustrate the interpretability of shapelets. With the rapid growth of the time series data, we plan to investigate a distributed shapelet discovery version of IPS and apply the IPS for multivairate TSC in the future.

Acknowledgments. We thank the anonymous reviewers for their helpful feedbacks. This work has been supported by the Hong Kong Research Grant Council (RGC), CRF C2004-21GF, RIF R2002-20, and GRFs HKBU12200021, 12202221. Dr. Li is partially supported by HKBU's Interdisciplinary Research Clusters Matching Scheme IRCMS/19-20/H01. This work was partially supported by the Health and Medical Research Fund from the Food and Health Bureau Commissioned Research on Hepatitis of the Hong Kong Government (Reference no: CID-CUHK-D) to Grace Wong.

REFERENCES

[1] S. Alaee, K. Kamgar, and E. Keogh. Matrix profile xxii: Exact discovery of time series motifs under dtw. *2020 IEEE 20th international conference on data mining (ICDM)*, 2020.

[2] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data mining and knowledge discovery*, 31(3):606–660, 2017.

[3] A. Bagnall, J. Lines, J. Hills, and A. Bostrom. Time-series classification with cote: the collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2522–2535, 2015.

[4] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[5] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[6] S. Cohen and Y. Matias. Spectral bloom filters. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 241–252, 2003.

[7] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004.

[8] H. A. Dau and E. Keogh. Matrix profile v: A generic technique to incorporate domain knowledge into motif discovery. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 125–134, 2017.

[9] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. The ucr time series classification archive, October 2018. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[10] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(Jan):1–30, 2006.

[11] Z. Fang, P. Wang, and W. Wang. Efficient learning interpretable shapelets for accurate time series classification. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 497–508, 2018.

[12] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 2019.

[13] J. C. B. Gamboa. Deep learning for time-series analysis. *arXiv preprint arXiv:1701.01887*, 2017.

[14] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *Proceedings of the VLDB Endowment*, volume 99, pages 518–529, 1999.

[15] M. Goswami, R. Pagh, F. Silvestri, and J. Sivertsen. Distance sensitive bloom filters without false negatives. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 257–269. SIAM, 2017.

[16] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme. Learning time-series shapelets. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 392–401, 2014.

[17] J. Grabocka, M. Wistuba, and L. Schmidt-Thieme. Fast classification of univariate and multivariate time series through shapelet discovery. *Knowledge and information systems*, 49(2):429–454, 2016.

[18] Y. He, X. Chu, and Y. Wang. Neighbor profile: Bagging nearest neighbors for unsupervised time series mining. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 373–384. IEEE, 2020.

[19] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.

[20] L. Hou, J. T. Kwok, and J. M. Zurada. Efficient learning of timeseries shapelets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1209–1215, 2016.

[21] M. Längkvist, L. Karlsson, and A. Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

[22] K. G. Larsen and J. Nelson. Optimality of the johnson-lindenstrauss lemma. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 633–638. IEEE, 2017.

[23] G. Li, B. Choi, J. Xu, S. S. Bhowmick, K. P. Chun, and G. Wong. Efficient shapelet discovery for time series classification. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[24] G. Li, B. Choi, J. Xu, S. S. Bhowmick, K. P. Chun, and G. Wong. Shapenet: A shapelet-neural network approach for multivariate time series classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[25] G. Li, B. Choi, J. Xu, S. S. Bhowmick, D. Mah, and G. Wong. Supplementary material of ips for time series classification, 2022. https://www.comp.hkbu.edu.hk/~csgzli/ips/ips-sup.pdf.

[26] J. Lines, L. M. Davis, J. Hills, and A. Bagnall. A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 289–297, 2012.

[27] A. Mueen, E. Keogh, and N. Young. Logical-shapelets: an expressive primitive for time series classification. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1154–1162, 2011.

[28] F. Pukelsheim. The three sigma rule. *The American Statistician*, 48(2):88–91, 1994.

[29] A. Rajaraman and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[30] T. Rakthanmanon and E. Keogh. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 668–676, 2013.

[31] M. Rhif, A. Ben Abbes, I. R. Farah, B. Martínez, and Y. Sang. Wavelet transform application for/in non-stationary time-series analysis: a review. *Applied Sciences*, 9(7):1345, 2019.

[32] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[33] J. Wang, Z. Wang, J. Li, and J. Wu. Multilevel wavelet decomposition network for interpretable time series analysis. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2437–2446, 2018.

[34] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

[35] L. Ye and E. Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 947–956, 2009.

[36] C.-C. M. Yeh, N. Kavantzas, and E. Keogh. Matrix profile iv: using weakly labeled time series to predict outcomes. *Proceedings of the VLDB Endowment*, 10(12):1802–1812, 2017.

[37] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh. Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 1317–1322, 2016.

[38] Y. Zhu, Z. Zimmerman, N. S. Senobari, C.-C. M. Yeh, G. Funning, A. Mueen, P. Brisk, and E. Keogh. Matrix profile ii: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 739–748. IEEE, 2016.